Computers Math. Applic. Vol. 27, No. 2, pp. 99–106, 1994
Printed in Great Britain.

0898-1221/94 $6.00 + 0.00

Key Generation of Algebraic-Code Cryptosystems

HUNG-MIN SUN
Institute of Computer Science and Information Engineering
National Chiao-Tung University, Hsinchu, Taiwan, Republic of China
TZONELIH HWANG
Institute of Information Engineering
National Cheng-Kung University, Tainan, Taiwan, Republic of China

(Received May 1992; revised and accepted November 1992)

Abstract—The purpose of this paper is to efficiently generate large nonsingular matrix (S, S^{-1}) pairs and permutation matrices over the binary field using short keys. The motivation of this work is to provide a solution to the long-key problem in algebraic-code cryptosystems. A special class of matrices which have exactly two 1's in each row and each column is defined, and their properties are investigated to facilitate the construction of these algorithms. The time complexities of these algorithms are studied and found to have $O(n)$ n-bit word operations.

Keywords—Algebraic-code cryptosystem, DBO matrices, DES, Private-key cryptosystem, Public-key cryptosystem.

1. INTRODUCTION

In 1978, McEliece proposed a public-key cryptosystem (McEliece's scheme) based on algebraic coding theory [1]. McEliece's scheme works as follows: the system user (receiver) constructs a $(k \times n)$ generator matrix $\mathbf{G}$ for a t-error correcting Goppa code $\mathbf{C}$, a $(k \times k)$ nonsingular matrix $\mathbf{S}$ over GF(2), and a random $(n \times n)$ permutation matrix $\mathbf{P}$. $\mathbf{G}$, $\mathbf{S}$, and $\mathbf{P}$ serve as secret keys of the receiver. Then, he computes $\mathbf{G}' = \mathbf{S}^{-1}\,\mathbf{G}\,\mathbf{P}^{-1}$, which is the generator matrix of a linear code (but supposedly hard to decode) with the same rate and error correction capability as $\mathbf{C}$. $\mathbf{G}'$ is published as the encryption key. The sender encrypts a k-bit message $\mathbf{m}$ into an n-bit ciphertext $\mathbf{c}$ by the equation $\mathbf{c} = \mathbf{m}\,\mathbf{G}' + \mathbf{e}$, where $\mathbf{e}$ is an n-bit random error vector of weight less than or equal to t, chosen by the sender. The receiver, knowing that $\mathbf{c}\ (= \mathbf{m}\,\mathbf{G}' + \mathbf{e} = \mathbf{m}\,\mathbf{S}^{-1}\,\mathbf{G}\,\mathbf{P}^{-1} + \mathbf{e})$, computes $\mathbf{c}\,\mathbf{P} = (\mathbf{m}\,\mathbf{S}^{-1})\,\mathbf{G} + \mathbf{e}\,\mathbf{P}$ and uses the decoding algorithm of the original code $\mathbf{C}$ to obtain the vector $\mathbf{m}\,\mathbf{S}^{-1}$. The plaintext can be recovered easily by $\mathbf{m} = (\mathbf{m}\,\mathbf{S}^{-1})\,\mathbf{S}$.

Rao and Nam modified the McEliece's scheme to construct a private-key algebraic-code cryptosystem (the Rao-Nam scheme) [2]. In this approach, $\mathbf{G}$, $\mathbf{S}$, $\mathbf{P}$, and $\mathbf{G}'$ are all kept secret. The Rao-Nam scheme performs encryption by the equation $\mathbf{c} = (\mathbf{m}\,\mathbf{S}^{-1}\,\mathbf{G} + \mathbf{e})\,\mathbf{P}^{-1}$, where $\mathbf{e}$ is a random error vector chosen from a predetermined syndrome-error table [2].

Both public-key and private-key algebraic-code cryptosystems require large binary matrices as keys. For example, the McEliece's scheme suggested the use of a (524 × 524) nonsingular matrix, a (524 × 1024) generator matrix, and a (1024 × 1024) permutation matrix as keys. In the Rao-Nam scheme, a (64 × 64) nonsingular matrix, a (64 × 72) generator matrix, and a (72 × 72) permutation matrix were suggested. If these matrices are used directly as keys, over 2×10^6 bits

The authors of this paper wish to thank the anonymous referees for their useful comments and suggestions.

are required for each user in McEliece's scheme, and over 18×10^3 bits are needed for each pair of users in the Rao-Nam Scheme. However, these matrices can be specified by a short sequence of bits (called seed or key seed).

While too short a key cannot provide security, a long key (as required for algebraic-code cryptosystems) is rather cumbersome and needs large storage space. Moreover, a long key does not necessarily provide a high level of security. There may be shortcuts which allow successful cryptanalysis in much less time than is required by exhaustive search on the key space [3]. In order to make algebraic-code cryptosystems (both public-key and private-key) more practical, the long-key problem has to be solved.

As a standard for private-key cryptosystems, the Data Encryption Standard (DES) uses a 56-bit key [4]. However, it is argued that with the advances in technology the key size of DES may soon have to be increased to 112 bits [5]. If we assume that a key size of 100 bits is appropriate, then in algebraic-code cryptosystems, some efficient algorithms are required to generate a binary matrix key set from a short key seed, e.g., to generate a matrix key set of size nearly 2^{100} from a 100-bit key seed.

An intuitive method is to use a data compression technique to compress these key matrices into short keys. However, a generalized data compression scheme cannot control the length of these short keys and, besides, the result of this compression is usually larger than what is needed. For example, the total number of 1024×1024 permutation matrices is 1024. The shortest sequence to represent a 1024×1024 permutation matrix is at least $\log_2(1024!)$ bits, where $\log_2(1024!) = \log_2 1 + \log_2 2 + \cdots + \log_2 1024 \geq \int_1^{1024} \log_2 x \, dx = 9215$, which is too large to be a key.

In these algebraic-code cryptosystems, both the nonsingular and permutation matrices are all held in secret. Therefore, even if the structure of these matrices reveals the key seed, there is no harm to the security of the system. The simplicity and efficiency of the algorithms will be our main concern.

2. DOUBLE-ONE (DBO) MATRICES AND THEIR PROPERTIES

DEFINITION 2.1. *An $n \times n$ square matrix over* GF(2) *is called a double-one* (DBO) *matrix if each column and each row of the matrix contains exactly two 1's.*

DEFINITION 2.2. *A double-one matrix is called* TYPE 1 *double-one* (DBO-1) *matrix if all 1's in the matrix can be connected in a unique cycle in either column or row direction (see Figure 1).*

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Figure 1. A DBO-1 matrix for $n = 4$; all 1's for a cycle: $(1,1) \rightarrow (3,1) \rightarrow (3,4) \rightarrow (4,4) \rightarrow (4,2) \rightarrow (2,2) \rightarrow (2,3) \rightarrow (1,3) \rightarrow (1,1)$, where (i,j) denotes the entry of the i^{th} row and the j^{th} column of the matrix.

DEFINITION 2.3. *A double-one matrix is called* TYPE 2 *double-one* (DBO-2) *matrix if it is not a* DBO-1 *matrix (see Figure 2).*

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2. A DBO-2 matrix for $n = 4$; all 1's form two cycles: $(1,1) \rightarrow (4,1) \rightarrow (4,3) \rightarrow (1,3) \rightarrow (1,1)$ and $(2,2) \rightarrow (3,2) \rightarrow (3,4) \rightarrow (2,4) \rightarrow (2,2)$.

DEFINITION 2.4. *The distance of two matrices of* GF(2) *with the same size is the number of entries with different values in the corresponding positions of both matrices.*

For example:

$$\text{The distance of } \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \text{ is 4.}$$

LEMMA 1. DBO-1 *matrices exist for* $n \geq 2$ *and* DBO-2 *matrices exist for* $n \geq 4$.

PROOF. According to Definition 1, it is easy to find that the only DBO-1 matrix for $n = 2$ is $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$.

For a DBO-1 matrix, there exists at least two cycles. Every cycle contains at least four 1's. So, the smallest size n of a DBO-2 matrix is $2 \cdot 4/2 = 4$. ∎

LEMMA 2. *For a square matrix of size* n, *the total number of* DBO-1*matrices is* $\frac{n}{2}[(n-1)!]^2$.

PROOF. Starting from the 1st row, assume i and j positions are selected for 1's. Obviously, there are C_2^n choices. Next, we select one position from the i^{th} column to put another "1" such that the i^{th} column has double 1's. There are $n-1$ choices. Suppose (k, i) is the position selected. Then, we need to select one position from the k^{th} row to put another "1" such that the k^{th} row has double 1's. There are $n-2$ choices. Repeat this process continuously until a DBO-1 matrix is obtained. The total number of ways to obtain a DBO-1 matrix can be computed by:

$$C_2^n \times (n-1) \times (n-2) \times (n-2) \times (n-3) \times (n-3) \times \cdots \times 2 \times 2 \times 1 \times 1 \times 1 = \frac{n}{2}[(n-1)!]^2. \quad ∎$$

LEMMA 3. *All* DBO *matrices are singular matrices.*

PROOF. For all $n \times n$ DBO matrices, if we add the first $n-1$ rows to the last row, then the entries of the last row are all 0's. This is because there are exactly two 1's in each column. ∎

LEMMA 4. *The rank of any* $n \times n$ DBO-1 *matrix is* $n-1$.

PROOF. Suppose $\mathbf{M}$ is an $n \times n$ DBO-1 matrix. For any $(x_1, \ldots, x_n)$ in its null space, we have

$$\mathbf{M} \times (x_1, \ldots, x_n)^{\top} = (0, \ldots, 0)^{\top}.$$

It is easy to compute that $(1, \ldots, 1)$ and $(0, \ldots, 0)$ are the only solutions for the above equation because all 1's in $\mathbf{M}$ form a cycle. For example,

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \longrightarrow \begin{cases} x_1 + x_3 = 0, \\ x_2 + x_3 = 0, \\ x_1 + x_2 = 0. \end{cases}$$

$$\text{If } x_1 = 1, \text{ then } x_3 = 1, \text{ and then } x_2 = 1.$$
$$\text{If } x_1 = 0, \text{ then } x_3 = 0, \text{ and then } x_2 = 0.$$

Therefore, $\{(0, \ldots, 0), (1, \ldots, 1)\}$ is the null space and its dimension is 1. By an important result in linear algebra for any $n \times n$ matrix,

$$\dim \text{(row space)} + \dim \text{(null space)} = n,$$

we obtain that the rank of $\mathbf{M}$ is $n-1$. ∎

THEOREM 5. *Adding one "1" to any entry of an $n \times n$ DBO-1 matrix, the resulting matrix is a nonsingular matrix of rank n.*

PROOF. Suppose $\mathbf{M}$ is the DBO-1 matrix and $\mathbf{M}^*$ is the resulting matrix by adding "1" to the entry (j,k) of $\mathbf{M}$. Consider the linear system:

$$\mathbf{A} \times (x_1, \ldots, x_n)^\top = (0, \ldots, 0)^\top,$$

where the coefficient matrix $\mathbf{A}$ is the resulting matrix by taking out the j^{th} row of $\mathbf{M}^*$. Similar to the proof of Lemma 4, we may obtain that $(0,\ldots,0)$, $(1,\ldots,1)$ are the only solutions for this system. Suppose the nonzero positions in the j^{th} row of $\mathbf{M}$ are (j,k_1), (j,k_2). Now, consider the system $\mathbf{M}^* \times (x_1,\ldots,x_n)^\top = (0,\ldots,0)^\top$. It has one more condition $x_{k_1} + x_{k_2} + x_k = 0$ than $\mathbf{A} \times (x_1,\ldots,x_n)^\top = (0,\ldots,0)^\top$. Thus, the only possible solution is $x_{k_1} = x_{k_2} = x_k = 0$. Hence, $(0,\ldots,0)$ is the only solution for this system, and we obtain dim (null space of $\mathbf{M}^*$) and the rank of $\mathbf{M}^*$ is n. ∎

3. ALGORITHMS FOR LARGE NONSINGULAR MATRICES S AND $\mathbf{S}^{-1}$

Based on Theorem 5, we construct an algorithm to efficiently generate a large nonsingular matrix $\mathbf{S}$ from a relatively short key seed. The algorithm has a one-to-one mapping from the key to the nonsingular matrix.

ALGORITHM I. (Input: A seed-key k, the length of k, $|k| < 2n-4$, e.g., 100-bit.
Output: $\mathbf{S}$, an $n \times n$ nonsingular matrix.)

Step 1: The seed-key k is used to specify a linear pseudo-random number generator with a one-to-one mapping from k to random sequence (e.g., LFSR [6] to generate a random sequence of length $2n-2$ with 0's in the last two bits. (These random bits $r_1 r_2 \ldots r_{2n-2}$ will be used to specify the location of 1's in the DBO-1 matrix.)

Step 2: Starting from an $n \times n$ zero matrix, fill the entry (1,1) with a "1," and lock the 1^{st} row such that the entries of the row cannot be changed; let $(R_i, C_i)(0 \leq i \leq 2n-1)$ be the index of the i^{th} "1" filled in the matrix; $(R_0, C_0)=(1,1)$; let (row,col) denote the index of the most recent "1" added to the matrix; (row,col)=(1,1).

Step 3: Repeat i for i=1...$2n-2$

```
BEGIN
IF i is even THEN /* Add 1 to the row */
                  BEGIN
                     Invert the (r_i + 1)^th (Note 1+1=2) available
                     (unlocked) 0 (from left to right) in the R_{i-1}^th
                     row, and lock the row;
                     update (row,col);
                     (R_i, C_i)=(row,col);
                  END
                ELSE /* Add 1 to the column */
                  BEGIN
                     Invert the (r_i + 1)^th (Note 1+1=2) available
                     (unlocked) 0 (from top to down) in the C_{i-1}^th
                     column, and lock the column;
                     update (row, col);
                     (R_i, C_i)=(row,col);
                  END
END
```

Step 4: Unlock the first row, and invert the entry $(1, C_{2n-2})$; update (row,col); (R_{2n-1}, C_{2n-1})=(row,col).

Step 5: Calculate p=($\lfloor k/n \rfloor \bmod n$)+1; q=($k \bmod n$)+1, and add "1" into the entry (p, q) (note here 1+1=0). Note: k is the integer value of the key seed.)

From Steps 1–4 of Algorithm I, we can construct an $n \times n$ DBO-1 matrix. Step 5 adds one "1" to this matrix. According to Theorem 5, the resulting matrix is indeed a nonsingular matrix. This proves the correctness of Algorithm I.

Obviously, the time complexity of Algorithm I is dominated by Step 3, which can be done in linear time. The array (R_i, C_i) recording the index of 1's in the newly constructed matrix will be used to construct its inverse later.

LEMMA 6. *The distance of two distinct* DBO *matrices of the same dimension is at least 4.*

PROOF. Let $\mathbf{D}_1$ and $\mathbf{D}_2$ be two distinct DBO matrices. We can find that at least one entry (assume (i, j) entry) in both matrices has different values. In this case, the j^{th} column in $\mathbf{D}_1$ and $\mathbf{D}_2$ must have at least two entries with different values because each column has two 1's. Similarly, for the i^{th} row, excluding (i, j) entry, we can find another entry (assume (i, k) entry) with different values. Thus, the k^{th} column in $\mathbf{D}_1$ and $\mathbf{D}_2$ has at least two entries with different values because each column has two 1's.

Based on the above discussions, we have proved that the distance of $\mathbf{D}_1$ and $\mathbf{D}_2$ is at least 4.

∎

THEOREM 7. *Algorithm I has a one-to-one mapping from the seed-key k to the nonsingular matrix.*

PROOF. It is possible to find a pseudo-random number generator such that there exists a one-to-one mapping from the seed-key k ($|k|$-bit) to a random sequence ($2n-2$ bits with 0's in the last two bits) for $|k| < 2n-4$. The random sequence is used to specify the locations of 1's in the DBO-1 matrix as described in Algorithm I. Now, what we need to show is that there exists a one-to-one mapping from the random sequence to a nonsingular matrix.

One-to-many mapping is impossible because Algorithm I is a deterministic algorithm. Assume that there exists a many-to-one mapping from random sequences to a nonsingular matrix. Let R_1, R_2 be two distinct random sequences which map to the same nonsingular matrix as follows:

$$R_1 = r_i r_2 \ldots r_i \ldots r_{2n-2} \xrightarrow{\text{DBO-1}} \mathbf{D}_1 \xrightarrow{\text{add one "1"}} \mathbf{S}_1,$$
$$R_2 = r_1 r_2 \ldots r'_i \ldots r'_{2n-2} \xrightarrow{\text{DBO-1}} \mathbf{D}_2 \xrightarrow{\text{add one "1"}} \mathbf{S}_2,$$

where the i^{th} bit is the first distinct element in R_1 and R_2.

According to Algorithm I, the i^{th} bit controls the $(i+1)^{\text{th}}$ "1" filled in the row or column, which the i^{th} "1" is located. Thus, this row (or column) of $\mathbf{D}_1$ will be different from that of $\mathbf{D}_2$ because $r_i \neq r'_i$. Hence, $\mathbf{D}_1$ is not equal to $\mathbf{D}_2$. However, according to Lemma 6, any two distinct DBO-1 matrices with the same dimension have the distance of at least 4. Therefore, the distance between $\mathbf{S}_1$ and $\mathbf{S}_2$ is at least 2 (we change only one bit in $\mathbf{D}_1$ to get $\mathbf{S}_1$ and one bit in $\mathbf{D}_2$ to get $\mathbf{S}_2$). That is, $\mathbf{S}_1 \neq \mathbf{S}_2$. This is a contradiction. Thus, Algorithm I is a one-to-one mapping from the random sequences to nonsingular matrices. ∎

LEMMA 8. *If one "1" is added to the entry (p, q) of a* DBO-1 *matrix, then the entries of the q^{th} row of its inverse $\mathbf{S}^{-1}$ are all 1's.*

PROOF. Assume the q^{th} row vector of $\mathbf{S}^{-1}$ is $\langle q_1, q_2 \ldots q_q \ldots q_n \rangle$ and $\mathbf{S} = [\mathbf{S}_1, \mathbf{S}_2 \ldots \mathbf{S}_q \ldots \mathbf{S}_n]$ where $\mathbf{S}_i$ is the i^{th}column of $\mathbf{S}$. Since $\mathbf{S}^{-1} \cdot \mathbf{S} = \mathbf{I}$,

$$\langle q_1 \ldots q_q \ldots q_n \rangle \cdot [\mathbf{S}_1 \ldots \mathbf{S}_q \ldots \mathbf{S}_n] = \langle 0, 0 \ldots \underset{\uparrow q^{\text{th}}}{1} \ldots 0, 0 \rangle. \tag{1}$$

Except $\mathbf{S}_q$ that has either one or three 1's, each column vector of $\mathbf{S}$ has exactly two 1's. Therefore, $q_1 = q_2 = \cdots = q_n = 1$ is the only solution for (1). ∎

Due to the special structure in the DBO-1 matrix, $\mathbf{S}^{-1}$ can be computed easily as follows. The q^{th} row of $\mathbf{S}^{-1}$ can be decided from Lemma 8. Assume that the entry (k, m) and the entry (k, q) of $\mathbf{S}$ are the only locations of 1's in the k^{th} row. The m^{th} row of $\mathbf{S}^{-1}$ can be determined by the following:

$$\mathbf{S} \cdot \mathbf{S}^{-1} = I$$

$$\begin{matrix} k^{\text{th}} \\ \text{row} \rightarrow \end{matrix} \begin{bmatrix} 0\ldots10\ldots10\ldots0 \\ \uparrow \quad \uparrow \\ m^{\text{th}}\ q^{\text{th}} \end{bmatrix} \begin{matrix} m^{\text{th}} \text{ row} \rightarrow \\ \times \\ q^{\text{th}} \text{ row} \rightarrow \end{matrix} \begin{bmatrix} 1\ldots101\ldots1 \\ \uparrow\ k^{\text{th}} \\ 1111\ldots1111 \end{bmatrix} = \begin{bmatrix} 0\ldots10\ldots0 \\ \uparrow \\ k^{\text{th}} \end{bmatrix} \leftarrow \begin{matrix} k^{\text{th}} \\ \text{row} \end{matrix}$$

matrix $\mathbf{S}$ | matrix $\mathbf{S}^{-1}$ | identity matrix

Similarly, other rows of $\mathbf{S}^{-1}$ can be computed in this way. The following algorithm is constructed to generate $\mathbf{S}^{-1}$.

ALGORITHM II. (Input: $\mathbf{S}$, an $n \times n$ nonsingular matrix constructed by Algorithm I,
Output: $\mathbf{S}^{-1}$, the inverse matrix of $\mathbf{S}$.)

Step 1: Obtain the arrays $R=[R_0, R_1, \ldots, R_{2n-1}], C=[C_0, C_1, \ldots, C_{2n-1}]$ from Algorithm I; search the array $\mathbf{C}$ to find the $1^{\text{st}} j$ such that $C_j=q$, $0 \le j \le 2-n-1$.

Step 2: IF $R_j=p$ THEN
BEGIN
$\mathbf{W}=[W_1, W_2, \ldots, W_n]$ where $W_i=R_{j+2\cdot i \bmod 2n}$
$\mathbf{L}=[L_1, L_2, \ldots, L_n]$ where $L_i=C_{j+2\cdot i \bmod 2n}$
END;
ELSE
BEGIN
$\mathbf{W}=[W_1, W_2, \ldots, W_n]$ where $W_i=R_{j+1-2\cdot i \bmod 2n}$
$\mathbf{L}=[L_1, L_2, \ldots, L_n]$ where $L_i=C_{j+1-2\cdot i \bmod 2n}$
END;

Step 3: Search the array $\mathbf{W}$ to find the m such that $W_m=p (1 \le m \le n)$; let $\mathbf{S}^{-1}$ be an $n \times n$ empty matrix; set the $L_1, \ldots, L_{m-1}$ rows of the matrix $\mathbf{S}^{-1}$ to 1's; set the $L_m, \ldots, L_n$ rows of the matrix $\mathbf{S}^{-1}$ to 0's.

Step 4: Repeat i for $i=1 \ldots N$
BEGIN
Invert the unlocked entries of the W_i^{th} column of $\mathbf{S}^{-1}$;
lock the L_i^{th} row.
END

It is easy to see that the time complexity is dominated by Step 3 which can be done in $O(n)$ n-bit word operations. In the following, we give an example to illustrate Algorithm II.

EXAMPLE 1.

$$k = 01011 \longrightarrow [\square] \longrightarrow r = 10011000$$

seed (5 bits) | random number generator | (8 bits)

$$k = 01011_2 = 11_{10}, \quad p = (\lfloor 11/5 \rfloor \bmod 5) + 1 = 3, \quad q = (11 \bmod 5) + 1 = 2$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

matrix $\mathbf{S}$ initial $\mathbf{S}^{-1}$ matrix $\mathbf{S}^{-1}$

$$\mathbf{R} = [R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9] = [1, 3, 3, 2, 2, 5, 5, 4, 4, 1]$$
$$\mathbf{C} = [C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9] = [1, 1, 2, 2, 4, 4, 3, 3, 5, 5]$$
$$\mathbf{W} = [W_1, W_2, W_3, W_4, W_5] = [R_4, R_6, R_8, R_0, R_2] = [2, 5, 4, 1, 3]$$
$$\mathbf{L} = [L_1, L_2, L_3, L_4, L_5] = [C_4, C_6, C_8, C_0, C_2] = [4, 3, 5, 1, 2]$$

4. ALGORITHM FOR LARGE PERMUTATION MATRICES

A permutation matrix that has exactly one "1" in each column and each row can also be obtained from a DBO-1 matrix $\mathbf{M}$ by inverting the even positions of 1's in the cycle of $\mathbf{M}$, counting from any position. It is obvious that the resulting matrix has exactly one "1" in each column and row. Therefore, the algorithm for the permutation matrices can be constructed by modifying Algorithm I as follows.

ALGORITHM III. (Input: a seed-key k,
Output: an $n \times n$ permutation matrix.)

Step 1: The seed-key k is used to specify a pseudo-random number generator (e.g., LFSR) to generate a random sequence of length $n-1$ with a "0" in the last bit [6]. (These random bits $r_1 r_2 \ldots r_{n-1}$ will be used to specify the locations of 1's in the permutation matrix. If $|k| > n-2$, e.g., $|k|$=100, n=72, we can use multiple choices for r_i, e.g., let $0 \le r_i \le n-i$ such that a one-to-one mapping from k to random sequence is possible.)

Step 2: Starting from an $n \times n$ zero matrix, invert the entry (1,1); lock the 1st row; let (row,col) denote the index of the entry that is visited most recently; (row,col)=(1,1)

Step 3: Repeat i for i=1 ... $n-1$

BEGIN
Find the 1st available (unlock) entry (from top to down) in the (col)th column, and
lock the column;
update (row,col);
invert the $(r_i+1)^{\text{th}}$ (Note 1+1=2) available (unlocked) entry (from left to right) in the (row)th row, and
lock the row;
update (row,col).
END

Notice that the time complexity of Algorithm III is $O(n)$ n-bit word operations.

5. CONCLUSIONS

The conventional methods to obtain the inverse matrix of an $n \times n$ nonsingular matrix needs $O(n^2)$ vector operations [7,8]. Based on the newly defined class of matrices (the Double-One matrices), we construct algorithms for generating large nonsingular matrices pairs and permutation matrices from a short seed in $O(n)$ n-bit word operations. These algorithms provide a 1-1 mapping between the key values and the matrices. They are particularly useful in solving the long-key

problem of algebraic-code cryptosystems. For the public-key algebraic-code cryptosystems, the generator matrix $\mathbf{G}$ can be recomputed from the public key $\mathbf{G}'$ and the secret matrices $\mathbf{S}$ and $\mathbf{P}$, by $\mathbf{G} = \mathbf{S}\mathbf{G}'\mathbf{P}$ by the receiver. Therefore, one may not have to construct a $\mathbf{G}$ based on a short seed. However, the problem of specifying the generator matrix from a short key seed for private-key algebraic-code cryptosystems still requires further research.

REFERENCES

1. R.J. McEliece, A public-key cryptosystem based on algebraic coding theory, DSN Progress Report 42–44, 114–116, JPL, Pasadena, CA, (1978).
2. T.R.N. Rao and K.H. Nam, Private-key algebraic-code encryption, *IEEE Trans. Infor. Theory* **35** (4), 829–833 (July 1989).
3. W. Diffie and M.E. Hellman, Exhaustive crypt-analysis of the NBS data encryption standard, *Computer* **10** (6), 74–84 (June 1977).
4. Data encryption standard, FIPS PUB 46, National Bureau of Standards, Washington, DC, (January 1977).
5. M.E. Hellman, DES will be totally insecure within ten years, *IEEE Spectrum* **16** (7), 32–39 (July 1979).
6. S.W. Golumb, *Shift Register Sequences*, Holden-Day, San Francisco, CA, (1967).
7. A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, (1974).
8. T. Hwang, Secret error-correcting codes and algebraic-code cryptosystems, Ph.D. Dissertation, Univ. of SW Louisiana, (Summer 1988).